**Senior DS Modelling Task @ Monsoon**

**By: Gauranshu**

### **Data Loading and Understanding**

The data was distributed into three files. So, my first task was to understand the significance of each file and the features they contained that might influence the target value. Then I had to decide on a method to merge all three files together.

**train\_flag:**

It contained one single feature, **NAME\_CONTRACT\_TYPE**, and one column for the user **uid**s. **NAME\_CONTRACT\_TYPE** contained **two unique categories**, which could be one-hot encoded to make them useful.

**accounts\_data\_train.json:**

It contains details of loans taken before the application for a loan from our client. The features that it contains are:

* **credit\_type:** It is a nominal feature with **14 unique categories**. The testing data had a slightly different set of these categories, so I generated **dummy values** to compensate for the missing categories and ignored new categories as it would be impossible for the model to process them as they were absent in the training data. Moreover, there are more prominent features that can compensate for these abnormalities.
* **loan\_amount:** It is a numerical feature with some outliers which were dealt with during the normalization step.
* **amount\_overdue:** It is also a numerical feature with some outliers dealt with in the same way. After analyzing their distributions over different target values, it was apparent that they have some correlation with the target values.
* **open\_date**: It is an ordinal feature. It cannot be used directly as a feature for model building but can be used to sort the data or combined with other features.
* **closed\_date:** It is also an ordinal feature with some values missing because the loan wasn’t disbursed at the time of data recording. These values were dropped during the training period as it would require accurate assumptions to replace the null values. This feature could be used to measure the loan period.
* **payment\_hist\_string:** It contained month-by-month history for the loan, with a value for each month corresponding to the number of days payment was overdue. It also gives us a measure of the duration of the loan period, along with the overdue status.

**enquiry\_data\_train.json:**

It contained previous loan enquiries made by the applicant. The features it has are **enquiry\_type** and **enquiry\_amount**. Due to the sparse nature of **enquiry\_type** and the absence of data related to the success of the applications, I decided not to use this data for the current model.

### **Probabilistic Model**

The first approach I used was based on utilizing **loan\_amount** and **amount\_overdue** to calculate some metrics which I could analyze to differentiate between good and bad loans. I used the **ratio** of the total **amount\_overdue** to the total **loan\_amount** and the frequency of successful disbursements over unsuccessful ones. The **ROC AUC score** for this method was **0.5**, which indicates that the results were randomly generated.

**Assumption:** For the applicants with no past loan history, the training data suggested that they are more likely to be tagged as bad loans. So, I returned bad loans for every applicant with zero loan history, and this **improved the score by 0.02**.

### **EDA and Feature Engineering**

With the plan of using an RNN in mind, I had to think of combining two files, **train\_flag.csv** and **accounts\_data\_train.json**. I added all the entries in the JSON file as individual entries in the CSV while mapping **NAME\_CONTRACT\_TYPE** and **TARGET** with each entry. After converting datetime strings to datetime objects, I sorted the DataFrame with respect to **open\_date**, which provides a fair idea of the loan application’s timestamps.

Three new features were created:

* **duration\_in\_months**: Difference between **closed\_date** and **open\_date**.
* **duration\_payment\_hist:** Number of months as per the payment history string.
* **difference:** Difference between **duration\_in\_months** and **duration\_payment\_hist**.

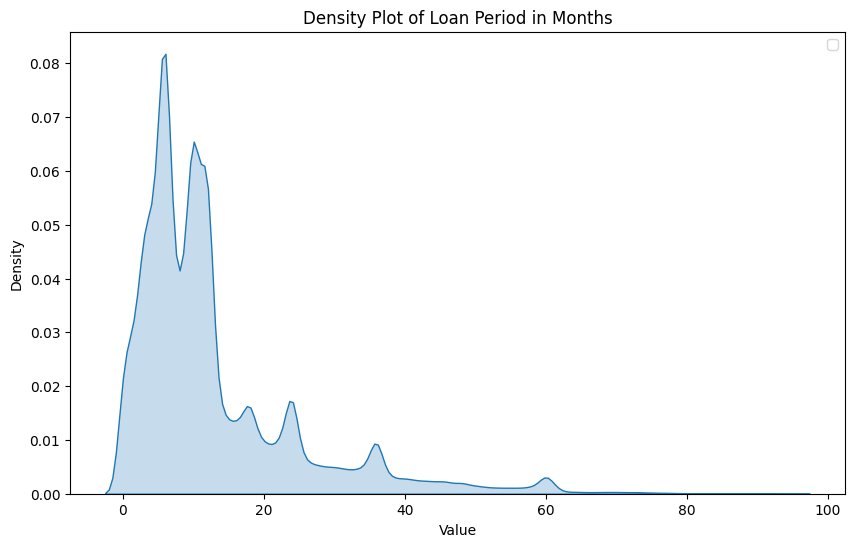
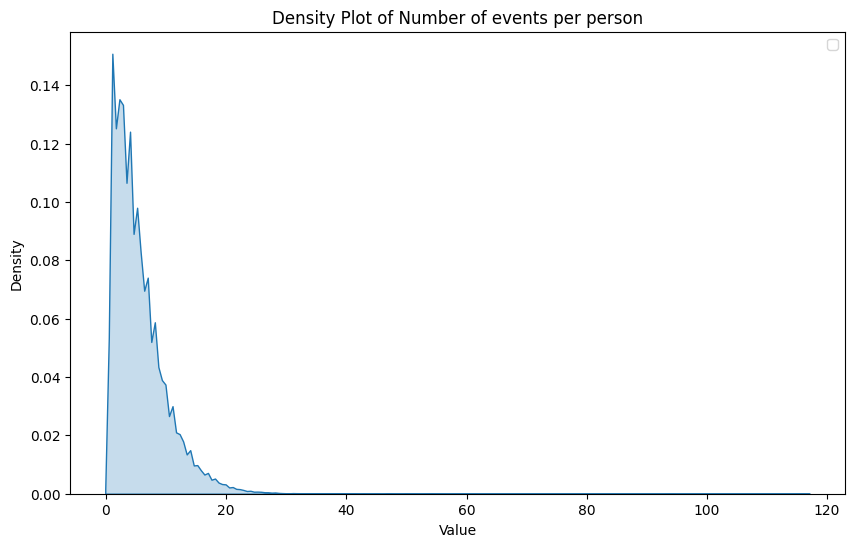
I made the following observations with the previously mentioned features:

* **duration\_in\_months** was negative in some cases because **closed\_date** came before **open\_date**. Some had reasons, while others might be mistakes. As such cases comprised only **0.2** percent of the total data, I dropped these entries.
* **difference** was non-zero in rare cases because some loans had amounts overdue after the closing date. This difference could act as a very important feature in determining bad loans.

I one-hot encoded the nominal features and normalized the numerical features except for the payment history. Because of the bounded values within payment history, outliers were not an issue. Moreover, payment history requires different processing steps. After dropping unnecessary columns, the DataFrame was grouped with respect to **uid**. These individual groups will serve as our individual data entries.

As we need to keep the size of every group and feature sizes consistent to convert them to torch.tensors, it is necessary to pad and clip the values. The only variable lengths for the current data are the group size and payment history.

Following are the plots for the group size (determined by the loan events per person) and loan duration as per the payment history, respectively. Both graphs are skewed toward lower values but extend far to the right as well.



To make the shape of the data consistent, I will keep payment history separate from other features. New loan events with every value as zero will be added to each group, appearing first with respect to time, until each group is of size 120. Additionally, 0 days overdue will be added to the beginning of the payment history to make every payment history of size 100.

Null entries in the features will enter first in the RNN and will be forgotten with successive iterations, dealing no effects on the final prediction.

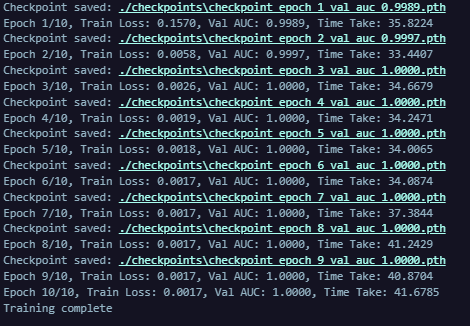
**RNN Model**

I used a batch size of 64, making the input shape for the features (64, 120, 22) and for the payment history (64, 120, 100). I embedded both of them using dense layers to the shape of (64, 120, 64) and (64, 120, 64), respectively. These embeddings were concatenated and passed to two consecutive RNN layers. The result, with a shape of (64, 1, 128), was passed to subsequent dense layers to compress the output into a binary answer. ReLU activation was used for all the layers, except for the final single-neuron layer, for which sigmoid activation was used instead.

**Training and Results**

Training is done with Adam optimizer scheduled with a learning rate of 10-4 with step size of 5. Binary cross entropy loss is used because of the nature of the problem and data. The model is trained for 10 epochs, saving checkpoints for the model that performed well on validation set w.r.t. AUC ROC score.

Following are the stats per epochs.



Following is confusion matrix for the predictions,

|  | Actually 0 | Actually |
| --- | --- | --- |
| Predicted 0 | 59891 | 21 |
| Predicted 1 | 3 | 3692 |

**Pipeline**

A pipeline was also developed to make inference on the data with given csv and json files.

**Scope for Improvement**

As per the confusion matrix, the model is predicting label 0 more confidently compared to label 1. This is after applying the appropriate threshold, but a class weight could be utilized here. I tested many architectures, and the final one was reported. However, I still feel that the architecture could be shrunken.

After a better understanding of the file **enquiry\_data\_train.json**, more features could be implemented into the model. The payment history could have utilized a separate shallow RNN layer to compress its results, which would help in shrinking the final architecture.